SQL

SQL is a special-purpose programming language designed for managing data held in a relational database management system.

In Oracle database:-

1. Rows = records and tuples.
2. Columns = attributes and Fields.
3. Table = Entity and relation.

The no. of uniques values in a column is called its “Cardinality”.

SQL Commands :-

Statement	Description	
Select	Retrieve data from a database	Data Manipulation Language (DML)
Insert	Insert a new rows in database	
Update	Change existing rows	
Delete	Remove unwanted rows	
Merge	Insert, Update and delete rows in single statement	
Create	Used to make data structure change in a database	Data Definition Language (DDL)
Alter		
Drop		
Rename		
Truncate		
Comment		

Grant	Grant and remove access to the oracle database	Data Control Language (DCL)
Revoke		
Commit		Transaction Control Language (TCL)
Rollback		
SavePoint		

Capabilities of SELECT statement :-

1. Selection :- Selecting columns from a table.
2. Projection :- Selecting rows from a table
3. Joining :- Fetching data from multiple database.

The Basics of SELECT Command :-

Select * [distinct] Columns , expression [alias].... From table;

The **select** and **from** are mandatory in a SQL command.

Select	Specifies the list of columns to be retrieved
*	Select all the columns
distinct	Removes all duplicates
columns	Name of columns returned by the query
expression	For ex. Sal + 3
alias	An alternate of the columns selected
From table	Specifies the table that contain those columns

SQL Data Types :-

Number(n,p) :- Exact number n with precision p.
CHAR(n) :- Fixed length character string.
VARCHAR(n) :- variable length character string.
DATE :- store year, month and day values.
BOOLEAN :- TRUE or FALSE.

DUAL table in SQL :-

To calculate the complex expressions that has nothing to do with the database or table, we use DUAL table provided by oracle. This has only one row and column.

Select (complex expression) from DUAL;

NULL Values :-

1. NULL values in oracle are shown as blank.
2. An expression with a NULL will be blank.
3. An operation with a NULL value and a character is not NULL.
4. NULL values can be selected using IS NULL.

Selection of data :-

Selection of rows is done by where clause. Condition will be applied under the where clause.

Column can be compared with another column.
Column can be compared with the expressions.
Column can be compared with a specified value.
Numeric column can be compared with another numeric column.
An expression can be put on both sides of the operator. Ex. a1/b1 = c1 *5.
Concatenation of a character and numeric value is character.
Characters comparison are case sensitive.
Character is specified in single quotes.

Date comparison - A date literal is also specified using the single quotes.
Default date format is **dd-mon-rr.**

Sorting :-

Order by clause is used to sort the data.

```
Syntax :- ORDER BY {col(s) | expr | numeric_pos}
                   [ASC|DESC]
                   [NULLS FIRST | LAST]
```

ORDER BY	Specifies the order in which rows are fetched and displayed
ASC / DESC	Rows will be sort ascending(default) or descending
NULL FIRST/LAST	Places the NULL values first or last after sorting all non null values

Order by can be done by using column names or position of column in the select statement.
Two or more columns can be used for sorting. This is called composite sorting.
The order of sorting would be left to right.

Comparison Operators :-

1. BETWEEN Operators :- Test whether a column falls within a range.
2. IN operator :- Test whether an item is member of a set of literal values.
3. LIKE operator :- Works for character data. Used for searching letters and words. LIKE can be used in 2 ways
    a. %(percentage) :- Eg. “S%” this will search for all the character data types that starts with S. W can use % after or before the word as well.
    b. _(underscore) :- Eg. “S_” this will search for characters that starts with S and have one more letter in it after S. Number of underscore followed corresponds to no. of letters.

SQL Functions :-

Components of a function :-

1. Input parameter list :- Zero or more arguments passed as input for processing.
2. Resultant value :- Upon execution only one value is returned by function.
3. Processing :- Program code that manipulates the input value, performs calculations and return the value.

Types of Function :-

Single Row Function :- Operate on one row of the dataset at a time.
Multiple Row Function :- Operate on more than one row at a time.

Character Functions :- There are 2 types of character functions.

1. **Case Conversion Functions** :- Convert the case of character data. From lower case to upper case or vice versa.
    a. **Lower** function :- Convert to lower case.
    b. **Upper** function :- Convert to upper case.
    c. **INITCAP** function :- Covert first letter of each word as upper and the rest as lower case.

    Case Conversion Function do not show an error when passed a numerical data.

2. **Character Manipulation Function** :-
    a. **CONCAT** function :- Join the 2 strings, expression or columns together. Ex :- concat(str1, str2) = str1str2. We can do concatenation using “||”. Ex:- str1 || str2.

b. **LENGTH** function :- return the length of the string. Length function also count the spaces in the string.

c. **LPAD** function :- Return a string padded with specified number of characters to the left. Ex :- lpad("Earth",10,'*') = "*****Earth". Also, lpad("Earth",3,'*') = Ear.

d. **RPAD** function :- Return a string padded with specified number of characters to the right. Ex :- rpad("Earth",10,'*') = "Earth*****". Also, rpad("Earth",3,'*') = Ear.

e. **TRIM** Function :-

Trim(string)	Removes white spaces from both side of string. Eg. trim(" hello ")= hello
Trim(trailing trimstring from s)	Removes all the occurrences of trimstring from the end of the string s if present. Ex :- trim (leading "x" from "xxhello") = hello
Trim(leading trimstring from s)	Removes all the occurrences of trimstring from the beginning of the string s if present.
Trim(both trimstring from s)	Removes all the occurrences of trimstring from the beginning and end of the string s if present. Ex :- trim(both "x" from "xxhelloxx") = hello.

- Trimstring can only have one letter.
- Trim function do not remove trimstring from the middle of the string.

f. **INSTR** function :- Locates the position of a search string within a given string.
INSTR(source, search string, [search start position], [nth occurrence])
Eg. instr(sysdate, 'MAY') = 4.

g. **SUBSTR** function :- Returns a subset from a given source string.
SUBSTR(source, search start position, [no of characters to extract])
substr("Ankit", 2,2) = nk.
substr("Ankit",-2,2) = it.

h. **REPLACE** function :- return the source string modified after replacing all the occurrence of a searchstring with a replacement string.
replace("Ankit","t","s") = Ankis.
If the replacement string item is missing then replace function will just remove Occurrence of all the searchstring from the source. Ex. replace("Ankit","n") = Akit.

Numerical Functions :- There are 3 numeric functions.

a. **ROUND** function :- performs a rounding operations on a numeric value based on the decimal precision specified.
Round(46.75,1) = 46.7
Round(46.75,0) = 47
Round(45.75,-1) = 50 as the first number before decimal is 5 so the number is round up to nearest 10.
Round(45.75,-2) = 0.

b. **TRUNC** function :- performs a truncation operations on a numeric value based on the decimal precision specified.
trunc(46.751,2) = 46.75
trunc(46.751,0) = 46
trunc(46.715,-2) = 0
trunc(1602.22,-3) = 1000. Truncation function truncate down the number.

c. **MOD function :-** returns the remainder of the division of 2 numbers.

Working with Dates :-

Dates are stored internally in a numeric format that supports the storage of century, year, month and day detail with time information such as hours, minutes and seconds.

Date Functions :-

a. **SYSDATE :-** It has no parameters and returns the current system date.

Date Arithmetic :-

Date1 - Date2 = Num1
Date1 - Num1 = Date2
Date1 = Date2 + Num1

b. MONTHS_BETWEEN :- returns the no. of months between two date values.
c. ADD_MONTHS :- returns a date after adding the a specified no. of months to a given date value.
d. NEXT_DAY :- returns a date when the next occurrence of a specified day of the week occurs. Ex. NEXT_DAY(Date1, ‘FRIDAY’).
e. LAST_DAY :- returns the date of the last day in the month a specified day belongs to.
f. ROUND :- performs a rounding operation on a value based on a specified date precision format. round(16-Feb-2012, ‘MONTH’) = 1-Mar-2012
g. to_date :- convert a string to a date. to_date(‘12-MAY-1993’ ‘DD-MM-YYYY’).
h. TRUNC :- performs a truncation operation on a value based on a specified date precision format.

Data Type Conversion :-

1. **Implicit Conversion** :- Values that do not share identical data types with function parameters are implicitly converted to the required format if possible. We don't have to apply a function to convert them.
   Ex. '1' + '2' = 3

2. **Explicit Conversion** :-
   a. TO_CHAR = returns an item of the data type VARCHAR2. Eg. - TO_CHAR(sal, $9999.99) will return a character representation of salary in dollar. This function works like "text" function in excel.
   b. TO_DATE = returns an item of type date. Eg. TO_DATE("25-MAY-2014","DD_MON_YYYY).
   c. TO_NUMBER = returns an item of type NUMBER. Eg. TO_NUMBER('$99.99","$99.99) = 99.99.
   d. TO_CHAR = returns an item of type CHAR.

General Functions :-

1. **NVL Function** :- It evaluates whether a column or expression of any data type is null or not. If the term is null, an alternate non null value would be returned; otherwise the initial term is returned. Eg. NVL(original, ifnull).
2. **NVL Function** :- It evaluates whether a column or expression of any data type is null or not. If the term is null, an alternate non null value would be returned; otherwise the third term is returned. Eg. NVL2(original, ifnull,elsenull).
3. **NULLIF Function** :- Tests two terms for equality. If they are equal the function returns a null, else it returns the first of the 2 values tested. Eg. NULLIF(1234,123+1) = 1234.
4. **COALESCE Function** :- Returns the first non-null value from its parameters list. If all its parameters are null, then null is returned.Eg. coalesce(substr('abc',4),"No String","No Value") = "No String".

Conditional Functions :-

1. **CASE Function** :-
```
CASE search_expr
        WHEN comparison_expr1 THEN iftrue 1
        WHEN comparison_expr2 THEN iftrue 2
        …
        WHEN comparison_expr3 THEN iftrue 3
        ELSE iffalse
        END;
```

2. Searched CASE function :- CASE

```
WHEN comparison_expr1 THEN iftrue 1
WHEN comparison_expr2 THEN iftrue 2
…
WHEN comparison_expr3 THEN iftrue 3
ELSE iffalse
END;
```

Group Functions :-

1. **COUNT Function :-** Execute on a column or expression and returns an integer that represents the number of rows in the group.
   Count(*) will count all the values including the null values.
   Count(column name) will ignore the null values in the column.
   Count(DISTINCT column name) will give the distinct no. of values in the column.

2. **SUM Function :-** Execute on numeric column and returns the sum of values in the group.

3. **AVERAGE Function :-** Returns the average of the columns ignoring the null values.

4. **MAX Function :-** Returns the largest value in the group.

5. **MIN Function :-** Returns the smallest value in the group.

GROUP BY Clause :-

This clause facilitates the creation of groups. It appears after the WHERE clause and before the ORDER BY clause.

Restricting the grouped data :- HAVING clause is used to restrict the group-level results. It is used after the group by and before the order by clause.
Ex:-

```
Select count(*) , dept_no from employee
Group by dept_no
Having count(*) > 3;
```